

On the Combinatorial Version of the Slepian–Wolf Problem

Daniyar Chumbalov and Andrei Romashchenko

Abstract

We study the following combinatorial version of the Slepian–Wolf coding scheme. Two isolated Senders are given binary strings X and Y respectively; the length of each string is equal to n , and the Hamming distance between the strings is at most αn . The Senders compress their strings and communicate the results to the Receiver. Then the Receiver must reconstruct both strings X and Y . The aim is to minimise the lengths of the transmitted messages.

For an asymmetric variant of this problem (where one of the Senders transmits the input string to the Receiver without compression) with deterministic encoding a nontrivial bound was found by A. Orlitsky and K. Viswanathany, [12]. In our paper we prove a new lower bound for the schemes with syndrome coding, where at least one of the Senders uses linear encoding of the input string.

For the combinatorial Slepian–Wolf problem with randomized encoding the theoretical optimum of communication complexity was found in [9], though effective protocols with optimal lengths of messages remained unknown. We close this gap and present a polynomial time randomized protocol that achieves the optimal communication complexity.

Index Terms

coding theory, communication complexity, pseudo-random permutations, randomized encoding, Slepian–Wolf coding

I. INTRODUCTION

The classic Slepian–Wolf coding theorem characterizes the optimal rates for the lossless compression of two correlated data sources. In this theorem the correlated data sources (two sequences of correlated random variables) are encoded separately; then the compressed data are delivered to the receiver where all the data are jointly decoded, see the scheme in Fig. 1. The

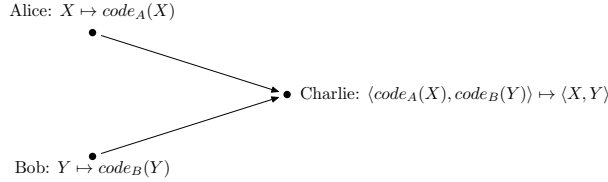


Fig. 1. Slepian–Wolf coding scheme

seminal paper [1] gives a very precise characterization of the profile of accessible compression rates in terms of Shannon’s entropies of the sources. Namely, if the data sources are obtained as $X = (x_1 \dots x_n)$ and $Y = (y_1 \dots y_n)$, where (x_i, y_i) , $i = 1, \dots, n$ is a sequence of i.i.d. random pairs, then all pairs of rates satisfying the inequalities

$$\begin{cases} |\text{Code}_A(X)| + |\text{Code}_B(Y)| & \geq H(X, Y) + o(n), \\ |\text{Code}_A(X)| & \geq H(X|Y) + o(n), \\ |\text{Code}_B(Y)| & \geq H(Y|X) + o(n), \end{cases}$$

can be achieved (with a negligible error probability); conversely, if at least one of the inequalities

$$\begin{cases} |\text{Code}_A(X)| + |\text{Code}_B(Y)| & \geq H(X, Y) - o(n), \\ |\text{Code}_A(X)| & \geq H(X|Y) - o(n), \\ |\text{Code}_B(Y)| & \geq H(Y|X) - o(n), \end{cases}$$

is violated, then the error probability becomes overwhelming. The areas of *achievable* and *non-achievable* rates are shown in Fig. 2 (the hatched green area consists of achievable points, and the solid red area consists of non-achievable points; the gap between these areas vanishes as $n \rightarrow \infty$).

A preliminary version of this paper (excluding Theorem 2 and Section 3) was presented at the MFCS 2015.

Daniyar Chumbalov is with Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland, Email: daniyar.chumbalov@epfl.ch

Andrei Romashchenko is with Laboratoire d’Informatique, de Robotique et de Microelectronique de Montpellier (LIRMM), France, Email: andrei.romashchenko@lirmm.fr

It is instructive to view the Slepian–Wolf coding problem in the general context of information theory. In the paper “*Three approaches to the quantitative definition of information*”, [13], Kolmogorov compared a *combinatorial* (cf. Hartley’s combinatorial definition of information, [14]), a *probabilistic* (cf. Shannon’s entropy), and an *algorithmic* approach (cf. Algorithmic complexity a.k.a. Kolmogorov complexity). Quite a few fundamental concepts and constructions in information theory have parallel implementations in all three approaches. A prominent example of this parallelism is provided by the formal information inequalities: they can be equivalently represented as linear inequalities for Shannon’s entropy, for Kolmogorov complexity, [15], or for (logs of) cardinalities of finite sets, [16], [17]. It is remarkable that many results known in one of these approaches look very similar to its homologues from the two other approaches, whereas the mathematical techniques and formal proofs behind them are fairly different.

As for the multi-source coding theory, two homologue theorems are known: the Slepian–Wolf coding theorem in Shannon’s framework (where the data sources are random variables, and the achievable rates are characterized in terms of the Shannon entropies of the sources) and Muchnik’s theorem on conditional coding, [18], in Kolmogorov’s framework (where the data sources are words, and the achievable rates are characterized in terms of the Kolmogorov complexities of the sources). What is missing in this picture is a satisfactory “combinatorial” version of the Slepian–Wolf theorem (though several partial results are known, see below). We try to fill this gap; we start with a formal definition of the combinatorial Slepian–Wolf coding scheme and then prove some bounds for the areas of achievable rates¹.

We focus on the binary symmetric case of the problem. In our (combinatorial) version of the Slepian–Wolf coding problem the data sources are binary strings, and the correlation between sources means that the Hamming distance between these strings is bounded. More formally, we consider a communication scheme with two senders (let us call them Alice and Bob) and one receiver (we call him Charlie). We assume Alice is given a string X and Bob is given a string Y . Both strings are of length n , and the Hamming distance between X and Y is not greater than a threshold αn . The senders prepare some messages $Code_A(X)$ and $Code_B(Y)$ for the receiver (i.e., Alice computes her message given X and Bob computes his message given Y). When both messages are delivered to Charlie, he should decode them and reconstruct both strings X and Y . Our aim is to characterize the optimal lengths of Alice’s and Bob’s messages.

This is the general scheme of the combinatorial version of the Slepian–Wolf coding problem. Let us place emphasis on the most important points of our setting:

- Alice knows X but not Y and Bob knows Y but not X ;
- one way communication: Alice and Bob send messages to Charlie without feedback;
- no communications between Alice and Bob;
- parameters n and α are known to all three parties.

In some sense, this is the “worst case” counterpart of the classic “average case” Slepian–Wolf problem.

It is usual for the theory of communication complexity to consider two types of protocols: deterministic communication protocols (Alice’s and Bob’s messages are deterministic functions of X and Y respectively, as well as Charlie’s decoding function) and randomized communication protocol (encoding and decoding procedures are randomized, and for each pair (X, Y) Charlie must get the right answer with only a small probability of error ε). In the next section we give the formal definitions of the deterministic and the randomized versions of the combinatorial Slepian–Wolf scheme and discuss the known lower and upper bounds for the achievable lengths of messages.

II. FORMALIZING THE COMBINATORIAL VERSION OF THE SLEPIAN–WOLF CODING SCHEME

In the usual terms of the theory of communication complexity, we study one-round communication protocols for three parties; two of them (Alice and Bob) send their messages, and the third one (Charlie) receives the messages and computes the final result. Thus, a formal definition of the communication protocol involves the coding functions for Alice and Bob and the decoding function for Charlie. We are interested not only in the total communication complexity (the sum of the lengths of Alice’s and Bob’s messages) but also in the trade-off between the two sent messages. In what follows we formally define two version of the Slepian–Wolf communication scheme — the deterministic and the probabilistic ones.

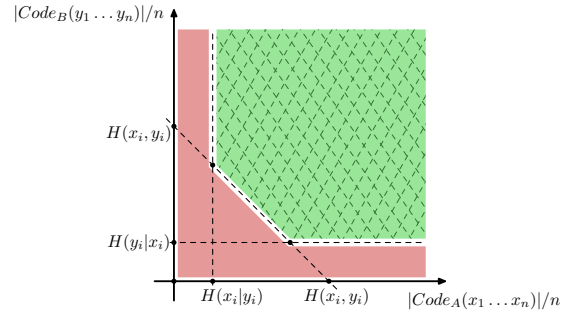


Fig. 2. The areas of achievable and non-achievable rates in the standard Slepian–Wolf theorem.

¹I. Csiszar and J. Körner described the Slepian–Wolf theorem as “*the visible part of the iceberg*” of the multi-source coding theory; since the seminal paper by Slepian and Wolf, many parts of this “iceberg” were revealed and investigated, see a survey in [19]. Similarly, Muchnik’s theorem has motivated numerous generalizations and extensions in the theory of Kolmogorov complexity. Apparently, a similar (probably even bigger) “iceberg” should also exist in the combinatorial version of information theory. However, before we explore this iceberg, we should understand the most basic multi-source coding models, and a natural starting point is the combinatorial version of the Slepian–Wolf coding scheme.

A. Deterministic communication schemes

In the deterministic framework the *communication protocol* for the combinatorial Slepian–Wolf coding scheme can be defined simply as a pair of uniquely decodable mappings — the coding functions of Alice and Bob.

Definition 1: We say that a pair of coding mappings

$$\begin{aligned} \text{Code}_A: \{0, 1\}^n &\rightarrow \{0, 1\}^{m_A} \\ \text{Code}_B: \{0, 1\}^n &\rightarrow \{0, 1\}^{m_B} \end{aligned}$$

is *uniquely decodable* for the combinatorial Slepian–Wolf coding scheme with parameters (n, α) , if for each pair of images $c_A \in \{0, 1\}^{m_A}$, $c_B \in \{0, 1\}^{m_B}$ there exist at most one pairs of strings (x, y) such that $\text{dist}(x, y) \leq \alpha n$, and

$$\begin{cases} \text{Code}_A(X) &= c_A, \\ \text{Code}_B(Y) &= c_B \end{cases}$$

(this means that the pair (X, Y) can be uniquely reconstructed given the values of $\text{Code}_A(X)$ and $\text{Code}_B(Y)$). If such a pair of coding mappings exists, we say that the pair of integers (m_A, m_B) (the lengths of the codes) is a pair of *achievable rates*.

If we are interested in effective constructions of the communication scheme, we can also explicitly introduce the decoding function for Charlie

$$\text{Decode}: (\text{Code}_A(X), \text{Code}_B(Y)) \mapsto (X, Y)$$

and investigate the computational complexities of these three mappings Code_A , Code_B and Decode .

We say that *encoding* in this scheme is *linear* (syndrome-coding), if both functions Code_A and Code_B in Definition 1 can be understood as linear mappings over the field of 2 elements:

$$\begin{aligned} \text{Code}_A: \mathbb{F}_2^n &\rightarrow \mathbb{F}_2^{m_A}, \\ \text{Code}_B: \mathbb{F}_2^n &\rightarrow \mathbb{F}_2^{m_B}. \end{aligned}$$

Further, we say that an encoding is *semi-linear*, if at least one of these two coding functions is linear.

B. Probabilistic communication schemes

We use the following standard *communication model with private sources of randomness*:

- each party (Alice, Bob, and Charlie) has her/his own “random coin” — a source of random bits (r_A , r_B , and r_C respectively),
- the coins are fair, i.e., produce independent and uniformly distributed random bits,
- the sources of randomness are private: each party can access only its own random coin.

In this model the message sent by Alice is a function of her input and her private random bits. Similarly, the message sent by Bob is a function of his input and his private random bits. Charlie reconstructs X and Y given both these messages and, if needed, his own private random bits. (In fact, in the protocols we construct in this paper Charlie will not use his own private random bits. The same time, the proven lower bounds remain true for protocols where Charlie employs randomness.) Let us give a more formal definition.

Definition 2: A randomized protocol for the combinatorial Slepian–Wolf scheme with parameters (n, α, ε) is a triple of mappings

$$\begin{aligned} \text{Code}_A: \{0, 1\}^n \times \{0, 1\}^R &\rightarrow \{0, 1\}^{m_A}, \\ \text{Code}_B: \{0, 1\}^n \times \{0, 1\}^R &\rightarrow \{0, 1\}^{m_B}, \end{aligned}$$

and

$$\text{Decode}: \{0, 1\}^{m_A+m_B} \times \{0, 1\}^R \rightarrow \{0, 1\}^n \times \{0, 1\}^n$$

such that for every pair of strings (x, y) satisfying $\text{dist}(x, y) \leq \alpha n$

$$\text{prob}_{r_A, r_B, r_C} [\text{Decode}(\text{Code}_A(X, r_A), \text{Code}_B(Y, r_B), r_C) = (X, Y)] > 1 - \varepsilon. \quad (1)$$

Here m_A is the length of Alice’s message and m_B is the length of Bob’s message. The second argument of the mappings Code_A , Code_B , and Decode should be understood as a sequence of random bits; we assume that each party of the protocol uses at most R random bits (for some integer R). Condition (1) means that for each pair of inputs $(X, Y) \in \{0, 1\}^n \times \{0, 1\}^n$ satisfying $\text{dist}(x, y) \leq \alpha n$, the probability of the error is less than ε .

When we discuss *efficient* communication protocols, we assume that the mappings Code_A , Code_B , and Decode can be computed in time polynomial in n (in particular, this means that only $\text{poly}(n)$ random bits can be used in the computation).

There is a major difference between the classic probabilistic setting of the Slepian–Wolf coding and the randomized protocols for combinatorial version of this problem. In the probabilistic setting we minimize the *average* communication complexity (for *typical* pairs (X, Y)); and in the combinatorial version of the problem we deal with the *worst* case communication complexity (the protocol must succeed with high probability for *each* pair (X, Y) with bounded Hamming distance).

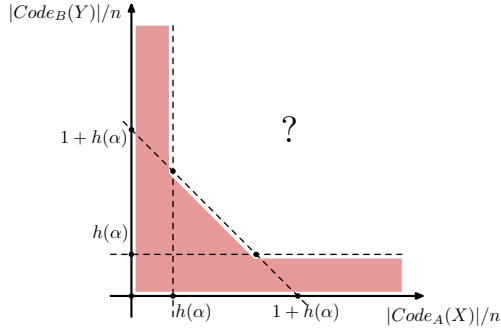


Fig. 3. The area of non-achievable rates.

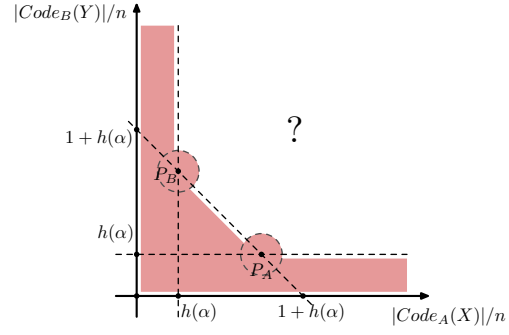


Fig. 4. Non-achievable rates for deterministic encoding.

C. The main results

A simple counting argument gives very natural lower bounds for lengths of messages in the deterministic setting of the problem:

Theorem 1 ([9]): For all $0 < \alpha < 1/2$, a pair (m_A, m_B) can be an achievable pair of rates for the deterministic combinatorial Slepian–Wolf problem with parameters (n, α) *only if* the following three inequalities are satisfied

- $m_A + m_B \geq (1 + h(\alpha))n - o(n)$,
- $m_A \geq h(\alpha)n - o(n)$,
- $m_B \geq h(\alpha)n - o(n)$,

where $h(\alpha)$ denotes Shannon’s entropy function,

$$h(\alpha) := -\alpha \log \alpha - (1 - \alpha) \log(1 - \alpha).$$

Remark 1: The proof of Theorem 1 is a straightforward counting argument. Let us observe that the bound $m_A + m_B \geq (1 + h(\alpha))n - o(n)$ (the lower bound for the total communication complexity) remains valid also in the model where Alice and Bob can communicate with each other, and there is a feedback from Charlie to Alice and Bob (even if we do not count the bits sent between Alice and Bob and the bits of the feedback from Charlie).

The asymptotic version of these conditions is shown in Fig. 3: the points in the area below the dashed lines are *not achievable*. Notice that these bounds are similar to the classic Slepian–Wolf bounds, see Fig. 2. The correspondence is quite straightforward: in Theorem 1 the sum of lengths of two messages is lower-bounded by the “combinatorial entropy of the pair” $(1 + h(\alpha))n$, which is basically the logarithm of the number of possible pairs (X, Y) with the given Hamming distance; in the classic Slepian–Wolf theorem the sum of two channel capacities is bounded by the Shannon entropy of the pair. Similarly, in Theorem 1 the lengths of both messages are bounded by $h(\alpha)n$, which is the “combinatorial conditional entropy” of X conditional on Y or Y conditional on X , i.e., the logarithm of the maximal number of X ’s compatible with a fixed Y and vice-versa; in the standard Slepian–Wolf theorem the corresponding quantities are bounded by the two conditional Shannon entropies.

Though the trivial bound from Theorem 1 looks very similar to the lower bounds in the classic Slepian–Wolf theorem and in Muchnik’s conditional coding theorem, this parallelism cannot be extended further. In fact, the bound from Theorem 1 is not optimal (for the deterministic communication protocols). More specifically, we cannot achieve any pairs of code lengths in $\Theta(n)$ -neighborhoods of the points $(n, h(\alpha)n)$ and $(h(\alpha)n, n)$ (in the dashed circles around points P_A and P_B in Fig. 4. This negative result was proven² in [12], see also a discussion in [9]. Thus, the bound from Theorem 1 does not provide the exact characterization of the set of achievable pairs. Here we see a sharp contrast with the classic Slepian–Wolf coding.

In this paper we prove another negative result for all *linear* and even for all *semi-linear* encodings:

Theorem 2: For each real $\alpha \in (0, \frac{1}{4})$ there exists an $\alpha' \in (\alpha, \frac{1}{2})$ such that for all achievable pairs (m_A, m_B) for the semi-linear deterministic combinatorial Slepian–Wolf scheme with a distance α , it holds

- $m_A + m_B \geq (1 + h(\alpha'))n - o(n)$,
- $m_A \geq h(\alpha')n - o(n)$,
- $m_B \geq h(\alpha')n - o(n)$.

Moreover, the value of α' can be defined explicitly as

$$\alpha' := \frac{1 - \sqrt{1 - 4\alpha}}{2}. \quad (2)$$

²More technically, [12] concerns an asymmetric version of the Slepian–Wolf scheme; it proves a lower bound for the length of $\text{Code}_A(X)$ assuming that $\text{Code}_B(Y) = Y$. Though this argument deals with only very special type of schemes where $\text{Code}_B(Y) = Y$, it also implies some bound for the general Slepian–Wolf problem: we can conclude that small neighborhoods around the points $(n, h(\alpha)n)$ and $(h(\alpha)n, n)$ cannot be achieved, as it is shown in Fig. 4, see Proposition 8 in Appendix.

The geometrical meaning of this bound is shown in Fig. 5: the area of non-achievable pairs of rates for given α becomes greater than the trivial counting lower bound: to the trivially forbidden area from Theorem 1 (the light red area in the picture) we add a new stripe of forbidden points (the dark red area).

It is instructive to compare the known necessary and sufficient conditions for the achievable rates. If we plug some linear codes approaching the Gilbert–Varshamov bound in the construction from [9, theorem 2], we obtain the following proposition.

Proposition 1: For each real $\alpha \in (0, \frac{1}{2})$ there exists a function $\delta(n) = o(n)$ such that and for all n , all pairs of integers (m_A, m_B) satisfying

- $m_A + m_B \geq (1 + h(2\alpha))n + \delta(n)$,
- $m_A \geq h(2\alpha)n + \delta(n)$,
- $m_B \geq h(2\alpha)n + \delta(n)$

are achievable for the deterministic combinatorial Slepian–Wolf scheme with parameters (n, α) . Moreover, these rates can be achieved with some *linear* schemes (where encodings of Alice and Bob are linear).

In Fig. 6 we combine together the known upper and lower bounds: the points in the light red area are non-achievable (for any deterministic scheme) due to Theorem 1; the points in the dark red area are non-achievable (for linear and semi-linear deterministic scheme) by Theorem 2; the points in the hatched green area are achievable due to Proposition 1. The gap between the known sufficient and necessary conditions remains pretty large.

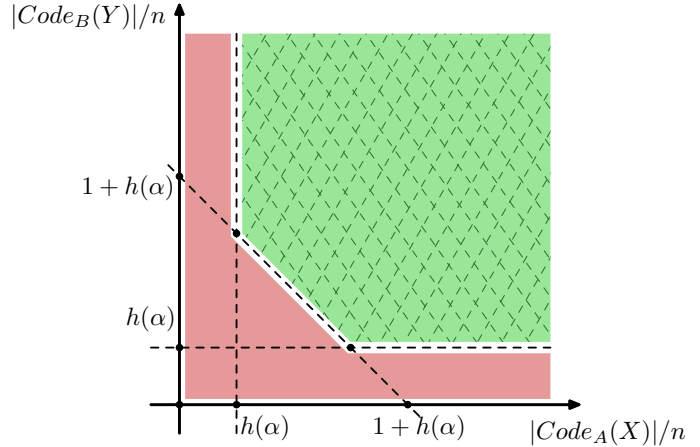
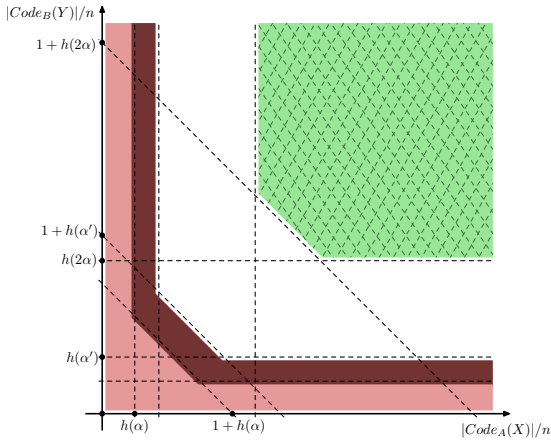


Fig. 6. Achievable and non-achievable rates for linear encoding. Fig. 7. The areas of achievable and non-achievable rates for randomized protocols.

Our proof of Theorem 2 (see Section III) is inspired by the classic proof of the Elias–Bassalygo bound in the coding theory, [21]. We do not know whether this bound holds for non-linear encodings. This seems to be an interesting question in between the coding theory and the communication complexity theory.

Thus, we see that the solution of the deterministic version of the combinatorial Slepian–Wolf problem is quite different from the standard Slepian–Wolf theorem. What about the probabilistic version? The same conditions as in Theorem 3 hold for the probabilistic protocols:

Theorem 3 ([9]): For all $\varepsilon \geq 0$ and $0 < \alpha < 1/2$, a pair (m_A, m_B) can be an achievable pair of rates for the probabilistic combinatorial Slepian–Wolf problem with parameters (n, α, ε) *only if* the following three inequalities are satisfied

- $m_A + m_B \geq (1 + h(\alpha))n - o(n)$,
- $m_A \geq h(\alpha)n - o(n)$,
- $m_B \geq h(\alpha)n - o(n)$,

Remark 2: The bounds from Theorem 3 holds also for the model with public randomness, where all the parties access a common source of random bits.

In the contrast to the deterministic case, for the probabilistic setting the sufficient conditions for achievable pairs are very close to the basic lower bound above. More precisely, for every $\varepsilon > 0$, all pairs in the hatched (green) area in Fig. 7 are achievable for the combinatorial Slepian–Wolf problem with parameters (n, α, ε) , see [9]. The gap between known necessary and sufficient conditions (the hatched and non-hatched areas in the figure) is negligibly small. Thus, for randomized protocols we get a result similar to the classic Slepian–Wolf theorem.

So, the case of randomized protocol for the combinatorial Slepian–Wolf problem seems closed: the upper and lower bounds known from [9] (asymptotically) match each other. The only annoying shortcoming of the result in [9] was computational complexity. The protocols in [9] require exponential computations on the senders and the receiver sides. In this paper we improve computational complexity of this protocol without degrading communication complexity. We propose a communication protocol with (i) optimal trade-off between the lengths of senders messages and (ii) polynomial time algorithms for all parties. More precisely, we prove the following theorem³:

Theorem 4: There exists a real $d > 0$ and a function $\delta(n) = o(n)$ such that for all $0 < \alpha < 1/2$ and all integers n , every pair (m_A, m_B) that satisfies three inequalities

- $m_A + m_B \geq (1 + h(\alpha))n + \delta(n)$,
- $m_A \geq h(\alpha)n + \delta(n)$,
- $m_B \geq h(\alpha)n + \delta(n)$,

is achievable for the combinatorial Slepian–Wolf coding problem with parameters $(n, \alpha, \varepsilon(n) = 2^{-\Omega(n^d)})$ (in the communication model with private sources of randomness). Moreover, all the computations in the communication protocol can be done in polynomial time.

Poly-time protocols achieving the marginal pairs $(n, h(\alpha)n + o(n))$ and $(h(\alpha)n + o(n), n)$ were originally proposed in [6] and later in [8]. We generalize these results: we construct effective protocols for all points in hatched area in Fig. 7. In fact, our construction uses the techniques proposed in [6] and [7].

The rest of this paper is organized as follows. In section III we discuss a non-trivial lower bound for communication complexity of the deterministic version of the combinatorial Slepian–Wolf coding scheme (a proof of Theorem 2). The argument employs binary Johnson’s bound, similarly to the proof of the well known Elias–Bassalygo bound in the coding theory.

In Section IV we provide an effective protocol for the randomized version of the combinatorial Slepian–Wolf coding scheme (a proof of Theorem 4). Our argument combines several technical tools: reduction of one global coding problem with strings of length n to many local problems with strings of length $\log n$ (similar to the classic technique of concatenated codes); Reed–Solomon checksums; pseudo-random permutations; universal hashing.

In conclusion we discuss how to make the protocol from Theorem 4 more practical — how to simplify the algorithms involved in the protocol. The price for this simplification is a weaker bound for the probability of error.

D. Notation

Through this paper, we use the following notation:

- we denote $h(\alpha) := \alpha \log(1/\alpha) + (1-\alpha) \log 1/(1-\alpha)$, and use the standard asymptotic bound for the binomial coefficients: $\binom{n}{\alpha n} = 2^{h(\alpha)n + O(\log n)}$,
- we denote by $\omega(x)$ the *weight* (number of 1’s) in a binary string x ,
- for a pair of binary strings x, y of the same length we denote by $x \oplus y$ their bitwise sum modulo 2,
- we denote by $\text{dist}(v, w)$ the Hamming distance between bit strings v and w (which coincides with $\omega(x \oplus y)$),
- For an n -bits string $X = x_1 \dots x_n$ and a tuple of indices $I = \langle i_1, \dots, i_s \rangle$ we denote $X_I := x_{i_1} \dots x_{i_s}$.

III. LOWER BOUNDS FOR DETERMINISTIC PROTOCOLS WITH SEMI-LINEAR ENCODING

In this section we prove Theorem 2. We precede the proof of this theorem by several lemmas. First of all, we define the notion of *list decoding* for the Slepian–Wolf scheme (similar to the standard notion of list decoding from the coding theory).

Definition 3: We say that a pair of coding mappings

$$\begin{aligned} \text{Code}_A: \{0, 1\}^n &\rightarrow \{0, 1\}^{m_A} \\ \text{Code}_B: \{0, 1\}^n &\rightarrow \{0, 1\}^{m_B} \end{aligned} \tag{3}$$

is *L-list decodable* for the combinatorial Slepian–Wolf coding scheme with parameters (n, α) , if for each pair of images $c_A \in \{0, 1\}^{m_A}$, $c_B \in \{0, 1\}^{m_B}$ there exist at most L pairs of strings (x, y) such that $\text{dist}(x, y) \leq \alpha n$, and

$$\begin{cases} \text{Code}_A(x) &= c_A, \\ \text{Code}_B(y) &= c_B. \end{cases}$$

The lengths of codewords of $\text{poly}(n)$ -decodable mappings must obey effectively the same asymptotical bounds as the codewords of uniquely decodable mappings. Let us formulate this statement more precisely.

Lemma 1: If (m_A, m_B) is an achievable pair of integers for the combinatorial Slepian–Wolf scheme with parameters (n, α) with list decoding (with the list size $L = \text{poly}(n)$), then

- $m_A + m_B \geq (1 + h(\alpha))n - o(n)$,
- $m_A \geq h(\alpha)n - o(n)$,

³Not surprisingly, the inequalities in Theorem 3 and Theorem 4 are very similar. The gap between necessary and sufficient conditions for achievable pairs is only $o(n)$.

- $m_B \geq h(\alpha)n - o(n)$.

The lemma follow from a standard counting argument. The lower bounds in this lemma are asymptotically the same as the bounds for the schemes with unique decoding in Theorem 3. The difference between the right-hand side of the inequalities in this lemma and in Theorem 3 is only $\log L$, which is negligible (an $o(n)$ -term) as $L = \text{poly}(n)$.

We will use the following well known bound from the coding theory.

Lemma 2 (Binary Johnson's bound): Let α and α' be positive reals satisfying (2). Then for every list of n -bits strings v_i ,

$$v_1, \dots, v_{2n+1} \in \{0, 1\}^n$$

with Hamming weights at most $\alpha'n$ (i.e., all v_i belong to the ball of radius $\alpha'n$ around 0 in Hamming's metrics), there exists a pair of strings v_i, v_j ($i \neq j$) such that

$$\text{dist}(v_i, v_j) \leq 2\alpha n.$$

Proof: See [20].

Now we are ready to prove the main technical lemma: every pair of mappings that is *uniquely* decodable for the Slepian–Wolf scheme with parameters (n, α) must be also $\text{poly}(n)$ -decodable with parameters (n, α') with some $\alpha' > \alpha$.

Lemma 3: Let α and α' be positive reals as in (2). If a pair of integers (m_A, m_B) is achievable for the combinatorial semi-linear Slepian–Wolf scheme with parameters (n, α) (with unique decoding), then the same pair is achievable for the combinatorial Slepian–Wolf scheme for the greater distance α' with $(2n)$ -list decoding. The value of α' can be explicitly defined from (2).

Proof: Let us fix some pair of encodings

$$\begin{aligned} \text{Code}_A: \{0, 1\}^n &\rightarrow \{0, 1\}^{m_A}, \\ \text{Code}_B: \{0, 1\}^n &\rightarrow \{0, 1\}^{m_B} \end{aligned}$$

that is *uniquely decodable* for pairs (x, y) with the Hamming distance αn . We assume that at least one of these mappings (say, Code_A) is linear. To prove the lemma we show that the same pair of encodings is *list poly}(n)*-list decodable for the pairs of strings with a greater Hamming distance $\alpha'n$.

Let us fix some $c_A \in \{0, 1\}^{m_A}$ and $c_B \in \{0, 1\}^{m_B}$, and take the list of all Code_A - and Code_B -preimages of these points:

- let $\{x_i\}$ be all strings such that $\text{Code}_A(x_i) = c_A$, and
- let $\{y_j\}$ be all strings such that $\text{Code}_B(y_j) = c_B$.

Our aim is to prove that the number of pairs (x_i, y_j) such that $\text{dist}(x_i, y_j) \leq \alpha'n$ is not greater than $2n$. Suppose for the sake of contradiction that the number of such pairs is at least $2n + 1$.

For each pair (x_i, y_j) that satisfy $\text{dist}(x_i, y_j) \leq \alpha'n$ we take their bitwise sum, $v := x_i \oplus y_j$. Since the Hamming distance between x_i and y_j is not greater than $\alpha'n$, the weight of v is not greater than $\alpha'n$. Thus, we get at least $2n + 1$ different strings v_s with Hamming weights not greater than $\alpha'n$. From Lemma 2 it follows that there exist a pair of strings v_{s_1}, v_{s_2} (say, $v_{s_1} = x_{i_1} \oplus y_{j_1}$ and $v_{s_2} = x_{i_2} \oplus y_{j_2}$) such that $\text{dist}(v_{s_1}, v_{s_2}) \leq 2\alpha n$. Hence, there exists a string w that is (αn) -close to both v_{s_1} and v_{s_2} , i.e.,

$$\text{dist}(v_{s_1}, w) \leq \alpha n \text{ and } \text{dist}(v_{s_2}, w) \leq \alpha n.$$

We use this w as a translation vector and define

$$z_1 := x_{i_1} \oplus w \text{ and } z_2 := x_{i_2} \oplus w.$$

For the chosen w we have

$$\text{dist}(z_1, y_{j_1}) = \omega(x_{i_1} \oplus w \oplus y_{j_1}) = \text{dist}(v_{s_1}, w) \leq \alpha n$$

and

$$\text{dist}(z_2, y_{j_2}) = \omega(x_{i_2} \oplus w \oplus y_{j_2}) = \text{dist}(v_{s_2}, w) \leq \alpha n.$$

Further, since $\text{Code}_A(x_{i_1}) = \text{Code}_A(x_{i_2}) = c_A$ and the mapping Code_A is linear, we get $\text{Code}_A(x_{i_1} \oplus w) = \text{Code}_A(x_{i_2} \oplus w)$. Hence,

$$\text{Code}_A(z_1) = \text{Code}_A(x_{i_1} \oplus w) = \text{Code}_A(x_{i_2} \oplus w) = \text{Code}_A(z_2).$$

Thus, we obtain two different pairs of strings (z_1, y_{j_1}) and (z_2, y_{j_2}) with the Hamming distances bounded by αn , such that

$$\begin{cases} \text{Code}_A(z_1) &= \text{Code}_A(z_2), \\ \text{Code}_B(y_{j_1}) &= \text{Code}_B(y_{j_2}). \end{cases}$$

This contradicts the assumption that the codes Code_A and Code_B are uniquely decodable for pairs at the distance αn . The lemma is proven.

Now we can prove Theorem 2. Assume that a pair of integers (m_A, m_B) is achievable for the combinatorial Slepian–Wolf coding scheme with unique decoding for a distance αn . From Lemma 3 it follows that the same pair is achievable for the combinatorial Slepian–Wolf coding scheme with $(2n)$ -list decoding with a greater distance $\alpha'n$. Then, we apply Lemma 1 and get the required bounds for m_A and m_B .

IV. RANDOMIZED POLYNOMIAL TIME PROTOCOL

A. Some technical tools

In this section we summarize the technical tools that we use to construct an effective randomized protocol.

1) Pseudo-random permutations

Definition 4: A distribution on the set S_n of permutations of $\{1, \dots, n\}$ is called *almost t -wise independent* if for every tuple of indices $1 \leq i_1 < i_2 < \dots < i_t \leq n$, the distribution of $(\pi(i_1), \pi(i_2), \dots, \pi(i_t))$ for π chosen according to this distribution has distance at most 2^{-t} from the uniform distribution on t -tuples of t distinct elements from $\{1, \dots, n\}$.

Proposition 2 ([5]): For all $1 \leq t \leq n$, there exists an integer $T = O(t \log n)$ and an explicit map $\Pi : \{0, 1\}^T \rightarrow S_n$, computable in time $\text{poly}(n)$, such that the distribution $\Pi(s)$ for random $s \in \{0, 1\}^T$ is almost t -wise independent.

2) Error correcting codes

Proposition 3 (Reed-Solomon codes): Assume $m + 2s < 2^k$. Then we can assign to every sequence of m strings $X = \langle X^1, \dots, X^m \rangle$ (where $X^j \in \{0, 1\}^k$ for each j) a string of *checksums* $Y = Y(X)$ of length $(2s + 1)k$,

$$Y : \{0, 1\}^{km} \rightarrow \{0, 1\}^{(2s+1)k}$$

with the following property. If at most s strings X^j are corrupted, the initial tuple X can be uniquely reconstructed given the value of $Y(X)$. Moreover, encoding (computation $X \mapsto Y(X)$) and decoding (reconstruction of the initial values of X) can be done in time $\text{poly}(2^k)$.

Proof: The required construction can be obtained from a systematic Reed-Solomon code with suitable parameters (see, e.g., [10]). Indeed, we can think of $X = \langle X^1, \dots, X^m \rangle$ as of a sequence of elements in a finite field $\mathbb{F} = \{q_1, q_2, \dots, q_{2^k}\}$. Then, we interpolate a polynomial P of degree at most $m - 1$ such that $P(q_i) = X_i$ for $i = 1, \dots, m$ and take the values of P at some other points of the field as checksums:

$$Y(X) := \langle P(a_{m+1}), P(a_{m+2}), \dots, P(a_{m+2s+1}) \rangle.$$

The tuple

$$\langle X^1, \dots, X^m, P(a_{m+1}), P(a_{m+2}), \dots, P(a_{m+2s+1}) \rangle$$

is a codeword of the Reed-Solomon code, and we can recover it if at most s items of the tuple are corrupted. It is well known that the error-correction procedure for Reed-Solomon codes can be implemented in polynomial time.

3) Universal hashing

Proposition 4 (universal hashing family, [11]): There exists a family of poly-time computable functions $\text{hash}_i : \{0, 1\}^n \rightarrow \{0, 1\}^k$ such that $\forall x_1, x_2 \in \{0, 1\}^n, x_1 \neq x_2$ it holds

$$\text{prob}_i[\text{hash}_i(x_1) = \text{hash}_i(x_2)] = 1/2^k,$$

where index i ranges over $\{0, 1\}^{O(n+k)}$ (i.e., each hash function from the family can be specified by a string of length $O(n+k)$ bits).

Such a family of hash functions can be constructed explicitly: the value of $\text{hash}_i(x)$ can be computed in polynomial time from x and i .

Parameter k in Proposition 4 is called *the length of the hash*.

The following claim is an (obvious) corollary of the definition of a universal hashing family. Let $\text{hash}_i(x)$ be a family of functions satisfying Proposition 4. Then for every $S \subset \{0, 1\}^n$, for each $x \in S$,

$$\text{prob}_i[\exists x' \in S, \text{ s.t. } x' \neq x \text{ and } \text{hash}_i(x) = \text{hash}_i(x')] < \frac{|S|}{2^k}.$$

This property allows to identify an element in S by its hash value.

4) The law of large numbers for t -independent sequences

The following version of the law of large numbers is suitable for our argument:

Proposition 5 (see [3], [4], [6]): Assume ξ_1, \dots, ξ_m are random variables ranging over $\{0, 1\}$, each with expectation at most μ , and for some $c < 1$, for every set of $t = m^c$ indices i_1, \dots, i_t we have

$$\text{prob}[\xi_{i_1} = \dots = \xi_{i_t} = 1] \leq \mu^t.$$

If $t \ll \mu m$, then

$$\text{prob} \left[\sum_{i=1}^m \xi_i > 3\mu m \right] = 2^{-\Theta(m^c)}.$$

More technically, we will use the following lemma:

Lemma 4: (a) Let ρ be a positive constant, $k(n) = \log n$, and $\delta = \delta(n)$ some function of n . Then for each pair of subsets $\Delta, I \subset \{1, \dots, k\}$ such that $|\Delta| = k$ and $|I| = \rho n$, for a k -wise almost independent permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$,

$$\mu := \text{prob}_\pi [|\pi(I) \cap \Delta| - \rho k| > \delta k] = O \left(\frac{1}{\delta^2 k} \right).$$

(b) Let $\{1, \dots, n\} = \Delta_1 \cup \dots \cup \Delta_m$, where Δ_j are disjoint sets of cardinality k (so $m = n/k$). Also we let $t = m^c$ (for some $c < 1$) and assume $t \ll \mu m$. Then, for a (tk) -wise almost independent permutation π ,

$$\begin{aligned} \text{prob}_\pi \left[\left| \pi(I) \cap \Delta_j \right| > (\rho + \delta)k \text{ for at least } 3\mu m \text{ different } j \right] &= 2^{-\Theta(m^c)}, \\ \text{prob}_\pi \left[\left| \pi(I) \cap \Delta_j \right| < (\rho - \delta)k \text{ for at least } 3\mu m \text{ different } j \right] &= 2^{-\Theta(m^c)}. \end{aligned}$$

(The proof is deferred to Section IV-F.)

Notice that a uniform distribution on the set of all permutation is a special case of a k -wise almost independent permutation. So the claims of Lemma 4 can be applied to a uniformly chosen random permutation.

B. Auxiliary communication models: shared and imperfect randomness

The complete proof of Theorem 4 involves a few different technical tricks. To make the construction more modular and intuitive, we split it in several possibly independent parts. To this end, we introduce several auxiliary communication models. The first two models are somewhat artificial; they are of no independent interest, and make sense only as intermediate steps of the proof of the main theorem. Here is the list of our communication model:

Model 1. The model with partially shared sources of perfect randomness: Alice and Bob have their own sources of independent uniformly distributed random bits. Charlie has a free access to Alice's and Bob's sources of randomness (these random bits are not included in the communication complexity); but Alice and Bob cannot access the random bits of each other.

Model 2. The model with partially shared sources of T -non-perfect randomness: Alice and Bob have their own (independent of each other) sources of randomness. However these sources are not perfect: they can produce T -independent sequences of bits and T -wise almost independent permutations on $\{1, \dots, n\}$. Charlie has a free access to Alice's and Bob's sources of randomness, whereas Alice and Bob cannot access the random bits of each other.

Model 3. The standard model with private sources of perfect randomness (our principal model). In this model Alice and Bob have their own sources of independent uniformly distributed random bits. Charlie cannot access random bits of Alice and Bob unless they include these bits in their messages.

We show that in all these models the profile of achievable pairs of rates is the same as in Theorem 3 (the hatched area in Fig. 7). We start with an effective protocol for Model 1, and then extend it to Model 2, and at last to Model 3.

C. An effective protocol for Model 1 (partially shared sources of perfect randomness)

In this section we show that all pairs of rates from the hatched area in 4 are achievable for Model 1. Technically, we prove the following statement.

Proposition 6: The version of Theorem 4 holds for the Communication Model 1.

Remark 1. Remark 1. Our protocol involves random objects of different kinds: randomly chosen permutations and random hash functions from a universal family. In this section we assume that the used randomness is perfect. This means that all permutations are chosen with the uniform distribution, and all hash functions are chosen independently.

1) Parameters of the construction

Our construction has some “degrees of freedom”; it involves several parameters, and values of these parameters can be chosen in rather broad intervals. In what follows we list these parameters, with short comments.

- λ is any fixed number between 0 and 1 (this parameter controls the ratio between the lengths of messages sent by Alice and Bob);
- κ_1, κ_2 (some absolute constants that control the asymptotic of communication complexity hidden in the $o(\cdot)$ -terms in the statements of Theorem 4 and Proposition 7);
- $k(n) = \log n$ (we will cut strings of Alice and Bob in “blocks” of length k ; we can afford the brute force search over all binary strings of length k , since 2^k is polynomial in n);
- $m(n) = n/k(n)$ (when we split n -bits strings into blocks of length k , we get m blocks);
- $r(n) = O(\log k) = O(\log \log n)$ (this parameter controls the chances to get a collision in hashing; we choose $r(n)$ so that $1 \ll r(n) \ll k$);
- $\delta(n) = k^{-0.49} = (\log n)^{-0.49}$ (the threshold for deviation of the relative frequency from the probability involved in the law of large numbers; notice that we choose $\delta(n)$ such that $\frac{1}{\sqrt{k}} \ll \delta(n) \ll k$);
- $\sigma = \Theta(\frac{1}{(\log n)^c})$ for some constant $c > 0$ (in our construction σn is the length of the Reed-Solomon checksum; we chose σ such that $\sigma \rightarrow 0$);
- t (this parameter characterize the quality of the random bits used by Alice and Bob; accordingly, this parameter is involved in the law(s) of large numbers used to bound the probability of the error; we let $t(n) = m^c$ for some $c > 0$).

2) The scheme of the protocol

Alice’s part of the protocol:

- (1_A) Select at random a tuple of λn indices $I = \{i_1, i_2, \dots, i_{\lambda n}\} \subset \{1, \dots, n\}$. Technically, we may assume that Alice chooses at random a permutation π_I on the set $\{1, 2, \dots, n\}$ and lets $I := \pi_I(\{1, 2, \dots, \lambda n\})$.
- (2_A) Send to the receiver the bits $X_I = x_{i_1} \dots x_{i_{\lambda n}}$, see Fig. 8.

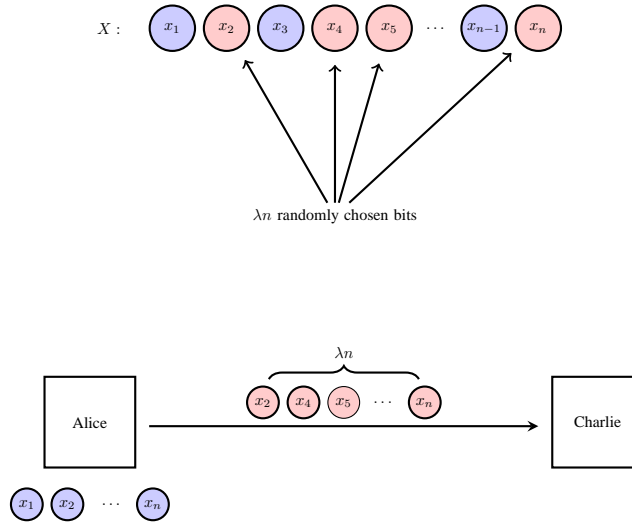


Fig. 8. Steps 1_A and 2_A: Alice selects λn bits in X and sends them to Charlie.

- (3_A) Choose another random permutation $\pi_A : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ and permute the bits of X , i.e., let⁴ $X' = x'_1 \dots x'_n := x_{\pi_A(1)} \dots x_{\pi_A(n)}$ (see Fig. 9). Further, divide X' into blocks of length $k(n)$, i.e., represent X' as a concatenation $X' = X'_1 \dots X'_m$, where $X'_j := x'_{(j-1)k+1} \dots x'_{jk}$ for each j (see Fig. 10).
- (4_A) Then Alice computes hash values of these blocks. More technically, we consider a universal family of hash functions

$$\text{hash}_l^A : \{0, 1\}^k \rightarrow \{0, 1\}^{h(\alpha)(1-\lambda)k + \kappa_1 \delta k + \kappa_2 \log k + r}.$$

With some standard universal hash family, we may assume that these hash functions are indexed by bit strings l of length $O(k)$, see Proposition 4. Alice chooses at random m indices l_1, \dots, l_m of hash functions. Then Alice applies each hash h_{l_j} to the corresponding block X'_j and sends to Charlie the resulting hash values $\text{hash}_{l_1}^A(X'_1), \dots, \text{hash}_{l_m}^A(X'_m)$, see Fig. 10.

⁴In what follows we consider also the π_A -permutation of bits in Y and denote it $Y' = y'_1 \dots y'_n := y_{\pi_A(1)} \dots y_{\pi_A(n)}$. Thus, the prime in the notation (e.g., X' and Y') implies that we permuted the bits of the original strings by π_A .

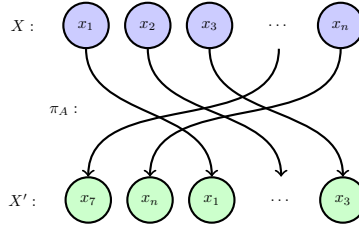


Fig. 9. Step 3_A: π_A permutes the bits of $X = x_1 x_2 \dots x_n$ and obtains X'

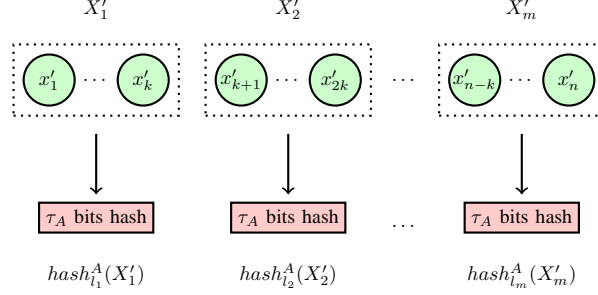


Fig. 10. Step 4_A: hashing the blocks of X' ; the length of each hash is equal to $\tau_A := h(\alpha)(1 - \lambda)k + \kappa_1 \delta k + \kappa_2 \log k + r$.

- (5_A) Compute the Reed-Solomon checksums of the sequence X'_1, \dots, X'_m that are enough to reconstruct all blocks X'_j if most σm of them are corrupted, and send them to Charlie. These checksums make a string of $O(\sigma m k)$ bits, see Proposition 3.
- (1_B) Choose at random a permutation $\pi_B : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ and use it to permute the bits of Y , i.e.,

Summary: Alice sends to Charlie three tranches of information,

- (i) λn bits of X selected at random,
- (ii) hashes for each of $m = n/k$ blocks in the permuted string X' ,
- (iii) the Reed-Solomon checksums for the blocks of X' .

Bob's part of the protocol:

- (1_B) Choose at random permutation $\pi_B : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ and use it to permute the bits of Y , i.e., let⁵ $Y'' = y''_1 \dots y''_n := y_{\pi_B(1)} \dots y_{\pi_B(n)}$, see Fig. 11.

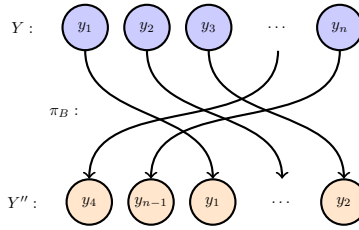


Fig. 11. Step 1_B: π_B permutes the bits of $Y = y_1 y_2 \dots y_n$ and obtains Y''

Further, divide Y'' into blocks of length k , and represent Y'' as a concatenation $Y'' = Y''_1 \dots Y''_m$, where $Y''_j := y''_{(j-1)k+1} y''_{(j-1)k+2} \dots y''_{jk}$ for each j , see Fig. 12.

- (2_B) Then choose at random m hash functions $\text{hash}_{l_j}^B$ from a universal family of hash functions

$$\text{hash}_l^B : \{0, 1\}^k \rightarrow \{0, 1\}^{(1-\lambda)k + h(\alpha)\lambda k + \kappa_1 \delta \cdot k + \kappa_2 \log k + r}.$$

(we assume that l_j are (T/k) -independent) and send to Charlie random hash values $\text{hash}_{l_1}^B(Y''_1), \dots, \text{hash}_{l_m}^B(Y''_m)$, see Fig. 12.

Similarly to (4_A), we may assume that these hash functions are indexed by bit strings l of length $O(k)$, see Proposition 4.

- (3_B) Compute the Reed-Solomon checksums of the sequence Y''_1, \dots, Y''_m , that are enough to reconstruct all blocks Y''_j , if at most σm of them are corrupted, and send them to Charlie. These checksums should be a string of length $O(\sigma m k)$ bits, see Proposition 3.

⁵Similarly, in what follows we apply this permutation to the bits of X and denote $X'' = x''_1 \dots x''_n := x_{\pi_B(1)} \dots x_{\pi_B(n)}$. Thus, the double prime in the notation (e.g., X'' and Y'') implies that we permuted the bits of the original strings by π_B .

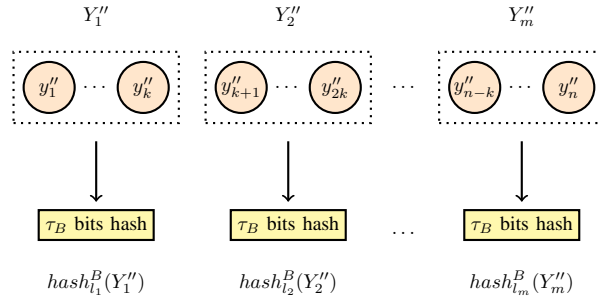


Fig. 12. Step 2_B : hashing the blocks of X' ; the length of each hash is equal to $\tau_B := (1 - \lambda)k + h(\alpha)\lambda k + \kappa_1 \delta \cdot k + \kappa_2 \log k + r$.

Summary: Bobs sends to Charlie two tranches of information,

- (i) hashes for each of $m = n/k$ blocks in the permuted string Y'' ,
- (ii) the Reed–Solomon checksums for the blocks of Y'' .

Charlie's part of the protocol:

- (1_C) Apply Bob's permutation π_B to the positions of bits selected by Alice, and denote the result by I'' , i.e.,

$$I'' = \{\pi_B(i_1), \dots, \pi_B(i_{\lambda n})\}.$$

Then split indices of I'' into m disjoint parts corresponding to the different intervals $Int_j = \{(j-1)k+1, (j-1)k+2, \dots, jk\}$, and $I''_j := I'' \cap Int_j$ (the red nodes in Fig. 13). Further, for each $j = 1, \dots, m$ denote by $X_{I''_j}$ the bits sent by Alice, that appear in the interval Int_j after permutation π_B . (We will show later that the typical size of $X_{I''_j}$ is close to λk .)

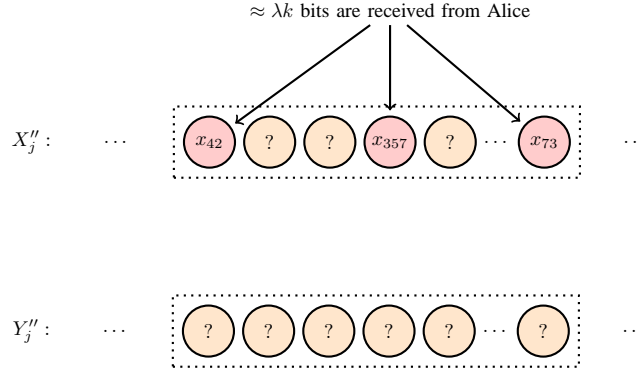


Fig. 13. Step 6_C : for each block X''_j Charlie typically gets from Alice $\approx \lambda k$ bits

- (2_C) For each $j = 1, \dots, m$ try to reconstruct Y''_j . To this end, find all bit strings $Z = z_1 \dots z_k$ that satisfy a pair of conditions (Cond₁) and (Cond₂) that we formulate below.

We abuse notation and denote by $Z_{I''_j}$ the subsequence of bits from Z that appear at the positions determined by I''_j . That is, if

$$I''_j = \{(j-1)k + s_1, \dots, (j-1)k + s_l\},$$

where

$$(j-1)k + s_1 < (j-1)k + s_2 < \dots < (j-1)k + s_l,$$

then $Z_{I''_j} = z_{s_1} z_{s_2} \dots z_{s_l}$. With this notation we can specify the required property of Z :

- (Cond₁) $\text{dist}(X_{I''_j}, Z_{I''_j}) \leq (\alpha + \delta)|I''_j|$, see Fig. 14,

- (Cond₂) $\text{hash}_{t_j}^B(Z)$ must coincide with the hash value $\text{hash}_{t_j}^B(Y''_j)$ received from Bob.

If there is a unique Z that satisfies these two conditions, then take it as a candidate for Y''_j ; otherwise (if there is no such Z or if there exist more than one Z that satisfy these conditions) we say that the procedure of reconstruction of Y''_j fails.

Remark 1: The requirement from (Cond₁) makes sense since in a typical case the indices from I'' are somehow uniformly distributed.

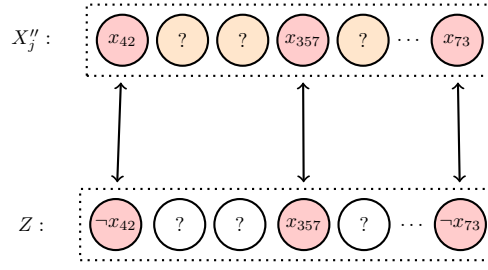


Fig. 14. Alice sends to Charlie $\approx \lambda k$ bits of the block X_j'' (red nodes); to construct a Z , Charlie inverts $\approx \alpha \cdot \lambda k$ bits of x_i in the “red” positions, and then choose the other bits arbitrarily.

Remark 2: There can be two kinds of troubles at this stage. First, for some blocks Y_j'' reconstruction fails (this happens when Charlie gets no or more than one Z that satisfy (Cond₁) and (Cond₂)). Second, for some blocks Y_j'' the reconstruction procedure seemingly completes, but the obtained result is incorrect (Charlie gets a Z which is not the real value of Y_j''). In what follows we prove that both events are rear. In a typical case, at this stage most (but not all!) blocks Y_j'' are correctly reconstructed.

- (3_C) Use Reed-Solomon checksums received from Bob to correct the blocks Y_j'' that we failed to reconstruct or reconstructed incorrectly at step (2_C).

Remark 3: Below we prove that in a typical case, after this procedure we get correct values of all blocks Y_j'' , so concatenation of these blocks gives Y'' .

- (4_C) Apply permutation π_B^{-1} to the bits of Y'' and obtain Y .
(5_C) Permute bits of Y and X_I using permutation π_A .
(6_C) For each $j = 1, \dots, m$ try to reconstruct X_j' . To this end, find all bit strings $W = w_1 \dots w_k$ such that
(Cond₃) at each position from $I' \cap \text{Int}_j$ the bit from X' (in the j -th block) sent by Alice coincides with the corresponding bit in W ,
(Cond₄) $\text{dist}(Y'_{\text{Int}_j \setminus I'_j}, W_{\text{Int}_j \setminus I'_j}) \leq (\alpha + \delta)|\text{Int}_j \setminus I'_j|$
(Cond₅) $\text{hash}_{I'_j}^A(W)$ coincides with the hash value $\text{hash}_{I'_j}^A(X'_j)$ received from Alice.

If there is a unique W that satisfies these conditions, then take this string as a candidate for X'_j ; otherwise (if there is no such W or if there exist more than one W satisfying these conditions) we say that reconstruction of X'_j fails.

Remark: We will show that in a typical case, most (but not all) blocks X'_j will be correctly reconstructed.

- (7_C) Use Reed-Solomon checksums received from Alice to correct the blocks X'_j that were incorrectly decoded at step (6_C).
Remark: We will show in a typical case, after this procedure we get correct values of all blocks X'_j , so concatenation of these blocks gives X' .
(8_C) Apply permutation π_A^{-1} to the positions of bits of X' and obtain X .

The main technical result of this section (correctness of the protocol) follows from the next lemma.

Lemma 5: In Communication Model 1, the protocol described above fails with probability at most $O(2^{-m^d})$ for some $d > 0$. (The proof is deferred to Section IV-F.)

3) Communication complexity of the protocol.

Alice sends λn bits at step (2_A), $h(\alpha)(1 - \lambda)k + O(\delta)k + O(\log k) + r$ for each block $j = 1, \dots, m$ at step (3_A), and σmk bits of the Reed-Solomon checksums at step (4_A). So the total length of Alice’s message is

$$\lambda n + (h(\alpha)(1 - \lambda)k + O(\delta)k + O(\log k) + r) \cdot m + \sigma n.$$

For the values of parameters that we have chosen above (see Section IV-C1), this sum can be estimated as $\lambda n + h(\alpha)(1 - \lambda)n + o(n)$. Bob sends $(1 - \lambda)k + h(\alpha)\lambda k + O(\delta)k + O(\log k) + r$ bits for each block $j = 1, \dots, m$ at step (1_B) and σmk bits of the Reed-Solomon checksums at step (2_B). This sums up to

$$((1 - \lambda)k + h(\alpha)\lambda k + O(\delta)k + O(\log k) + r) \cdot m + \sigma n$$

bits. For the chosen values of parameters this sum is equal to $(1 - \lambda)n + h(\alpha)\lambda n + o(n)$. When we vary parameter λ between 0 and 1, we variate accordingly the lengths of both messages from $h(\alpha)n + o(n)$ to $(1 + h(\alpha))n + o(n)$, whereas the sum of Alice’s and Bob’s messages always remains equal to $(1 + h(\alpha))n + o(n)$. Thus, varying λ from 0 to 1, we move in the graph in Fig. 7 from P_B to P_A .

It remains to notice that algorithms of all participants require only $\text{poly}(n)$ -time computations. Indeed, all manipulations with Reed-Solomon checksums (encoding and error-correction) can be done in time $\text{poly}(n)$, with standard encoding and decoding

algorithms. The brute force search used in the decoding procedure requires only the search over sets of size $2^k = \text{poly}(n)$. Thus, Proposition 6 is proven.

D. An effective protocol for Model 2

In this section we prove that the pairs of rates from Fig. 7 are achievable for Communication Model 2. Now the random sources of Alice and Bob are not perfect: the random permutations are only t -wise almost independent and the chosen hash functions are t -independent (for a suitable t).

Proposition 7: The version of Theorem 4 holds for Communication Model 2 (with parameter $T = \Theta(n^c \log n)$).

To prove Proposition 7 we do not need a new communication protocol — in fact, the protocol that we constructed for Model 1 in the previous section works for Model 2 as well. The only difference between Proposition 6 and Proposition 7 is a more general statement about the estimation of the error probability:

Lemma 6: For Communication Model 2 with parameter $T = \Theta(n^c \log n)$ the communication protocol described in section IV-C fails with probability at most $O(2^{-m^d})$ for some $d > 0$. (The proof is deferred to Section IV-F.)

Since the protocol remains the same, the bounds for the communication and computational complexity, proven in Proposition 6, remain valid in the new setting. With Lemma 6 we get the proof of Proposition 7.

E. The model with private sources of perfect randomness

Proposition 7 claims that the protocol from Section IV-C works well for the artificial Communication Model 2 (with non-perfect and partially private randomness). Now we want to modify this protocol and adapt it to Communication Model 3.

Technically, we have to get rid of (partially) shared randomness. That is, in Model 3 we cannot assume that Charlie access Alice's and Bob's random bits for free. Moreover, Alice and Bob cannot just send their random bits to Charlie (this would dramatically increase the communication complexity). However, we can use the following well-known trick: we require now that Alice and Bob use pseudo-random bits instead of truly uniformly random bits. Alice and Bob take short seeds for pseudo-random generators at random (with the truly uniform distribution) expand them to longer sequences of pseudo-random bits, and feed these *pseudo-random* bits in the protocol described in the previous sections. Alice and Bob transmit the random seeds of their generators to Charlie (the seeds are rather short, so they do not increase communication complexity substantially); so Charlie (using the same pseudo-random generators) expands the seeds to the same long pseudo-random sequences and plug them in into his side of the protocol.

More formally, we modify the communication protocol described in Section IV-C. Now Alice and Bob begin the protocol with the following steps:

(0_A) Alice choses at random the seeds for pseudo-random generators and send them to Charlie.

(0_B) Bob choses at random the seeds for pseudo-random generators and also send them to Charlie.

When these preparations are done, the protocol proceeds exactly as in Section IV-C (steps (1_A)–(5_A) for Alice, (1_B)–(3_B) for Bob, and (1_C)–(8_C) for Charlie). The only difference that all *random* objects (random hash functions and random permutations) are now *pseudo-random*, produces by pseudo-random generators from the chosen random seeds.

It remains to choose some specific pseudo-random generators that suits our plan. We need two different pseudo-random generators — one to generate indices of hash functions and another to generate permutations. Constructing a suitable sequence of pseudo-random hash-functions is simple. Both Alice and Bob needs m random indices l_i of hash functions, and the size of each family of hash functions is $2^{O(k)} = 2^{O(\log n)}$. We need the property of t -independency of l_i for $t = m^c$ (for a small enough c). To generate these bits we can take a random polynomial of degree at most $t - 1$ over $\mathbb{F}_{2^{O(\log n)}}$. The seed of this “generator” is just the tuple of all coefficients of the chosen polynomial, which requires $O(t \log n) = o(n)$ bits. The outcome of the generator (the resulting sequence of pseudo-random bits) is the sequence of values of the chosen polynomial at (some fixed in advance) m different points of the field. The property of t -independence follows immediately from the construction: for a randomly chosen polynomial of degree at most $t - 1$ the values at any t points of the field are independent.

The construction of a pseudo-random permutation is more involved. We use the construction of a pseudo-random permutation from [5]. We need the property of t -wise almost independence; by Proposition 2 such a permutation can be effectively produced by a pseudo-random generator with a seed of length $O(t \log n)$. Alice and Bob chose seeds for all required pseudo-random permutations at random, with the uniform distribution.

The seeds of the generators involved in our protocol are much shorter than n , so Alice and Bob can send them to Charlie without essentially increasing communication complexity.

The probability of the error remain the same is in Section IV-D, since we plugged in the protocol the pseudo-random bits which are T -wise independent. Hence, we can use the bound from Proposition 7. This concludes the proof of Theorem 4.

F. Proofs of the probabilistic lemmas

In this section we prove the technical probabilistic propositions used to estimate the probability of the failure in our communication protocols.

Proof of Lemma 4 (a): First we prove the statement for a uniformly independent permutations. Let $I = \{i_1, \dots, i_{\rho n}\}$. We denote

$$\xi_s = \begin{cases} 1, & \text{if } \pi(i_s) \in \Delta, \\ 0, & \text{otherwise.} \end{cases}$$

We use the fact that the variables ξ_s are “almost independent”. Since the permutation π is chosen uniformly, we have $\text{prob}[\xi_s = 1] = |\Delta|/n = k/n$ for each s . Hence, $\mathbb{E}(\sum \xi_s) = \rho k$. Let us estimate the variance of this random sum.

For $s_1 \neq s_2$ we have

$$\text{prob}[\xi_{s_1} = \xi_{s_2} = 1] = \frac{k}{n} \cdot \frac{k-1}{n-1} = \left(\frac{k}{n}\right)^2 + O(k/n^2).$$

So, the correlation between every two ξ_s is very weak. We get

$$\begin{aligned} \text{var}\left(\sum \xi_s\right) &= \mathbb{E}\left(\sum \xi_s\right)^2 - \left(\mathbb{E}\sum \xi_s\right)^2 \\ &= \sum_s \mathbb{E}\xi_s + \sum_{s_1 \neq s_2} \mathbb{E}\xi_{s_1}\xi_{s_2} - \left(\mathbb{E}\sum_s \xi_s\right)^2 = O(k). \end{aligned}$$

Now we apply Chebyshev’s inequality

$$\text{prob}_\pi\left[\left|\sum \xi_s - \rho k\right| > \delta k\right] < \frac{\text{var}(\sum \xi_s)}{(\delta k)^2} = O\left(\frac{1}{\delta^2 k}\right), \quad (4)$$

and we are done.

For a k -wise almost independent permutation we should add to the right-hand side of (4) the term $O(2^{-k})$, which does not affect the asymptotic of the final result.

Before we prove Lemma 4 (b), let us formulate a corollary of Lemma 4 (a).

Corollary 1: Let $\Delta_1, \dots, \Delta_t$ be some disjoint subsets in $\{1, \dots, n\}$ such that $|\Delta_j| = k$ for each j . Then for a uniformly random or (kt) -wise almost independent permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$,

$$\text{prob}_\pi\left[\left|\pi(I) \cap \Delta_j\right| > (\rho + \delta)k \text{ for all } j\right] \leq \mu^t$$

and

$$\text{prob}_\pi\left[\left|\pi(I) \cap \Delta_j\right| < (\rho - \delta)k \text{ for all } j\right] \leq \mu^t.$$

Proof of Corollary: (sketch) For a uniform permutation it is enough to notice that the events “there are too few π -images of I in Δ_j ” are negatively correlated with each other. That is, if we denote

$$E_j := \left\{\pi \mid \left|\pi(I) \cap \Delta_{j_1}\right| < (\rho - \delta(n))k\right\},$$

then

$$\text{prob}_\pi[E_1] > \text{prob}_\pi[E_{j_1} \mid E_2] > \dots > \text{prob}_\pi[E_{j_1} \mid E_2 \text{ and } E_2] > \dots$$

It remains to use the bound from (a) for the unconditional probabilities.

Similarly to the proof of Lemma 4 (a), in the case of almost independent permutations the difference of probabilities is negligible.

Proof of Lemma 4 (b): Follows immediately from the Corollary 1 and Proposition 5.

Proof of Lemma 5 and Lemma 6: We prove directly the statement of Lemma 6 (which implies of course Lemma 5). Let us estimate probabilities of errors at each step of Charlie’s part of the protocol.

Step (1_C): No errors.

Step (2_C): We should estimate probabilities of errors in reconstructing each block Y_j'' .

1st type error: the number of Alice’s bits $x_{i_1}, \dots, x_{i_{\lambda n}}$ that appear in the block Y_j'' is less than $(\lambda - \delta)k$. Technically, this event itself is not an error of decoding; but it is undesirable: we cannot guarantee the success of reconstruction of Y_j'' if we get in this slot too few bits from Alice. Denote the probability of this event (for a block $j = 1, \dots, m$) by p_1^B . By the law of large numbers, $p_1^B \rightarrow 0$ if $\delta \gg 1/\sqrt{k}$. This fact follows from Lemma 4(a).

2nd type error: 1st type error does not occur but

$$\text{dist}(X_{I_j''}, Y_{I_j''}) > (\alpha + \delta)|I_j''|.$$

Denote the probability of this event (for a block $j = 1, \dots, m$) by p_2^B . Again, by the law of large numbers, $p_2^B \rightarrow 0$ if $\delta \gg 1/\sqrt{k}$. Technically, we apply Lemma 4(a) with $\rho = \alpha$ and $\delta = 1/k^{0.49}$.

3rd type error: 1st and 2nd type errors do not occur but there exist at least two different strings Z satisfying (1) and (2). We choose the length of hash values for hash_l^B so that this event happens with probability less than $p_3^B = 2^{-r}$. Let us explain this in more detail.

All the positions of Int_j are split into two classes: the set I_j'' and its complement $\text{Int}_j \setminus I_j''$. For each position in I_j'' Charlie knows the corresponding bit from X'' sent by Alice. To get Z , we should

- (i) invert at most $(\alpha + \delta)|I_j''|$ of Alice's bits (here we use the fact that the 2nd type error does not occur), and
- (ii) choose some bits for the positions in $\text{Int}_j \setminus I_j''$ (we have no specific restrictions for these bits).

The number of all strings that satisfy (i) and (ii) is equal to

$$S_B := \sum_{s=0}^{(\alpha+\delta)|I_j''|} \binom{|I_j''|}{s} \cdot 2^{|\text{Int}_j \setminus I_j''|} = 2^{h(\alpha)\lambda k + (1-\lambda)k + O(\delta)k + O(\log k)}.$$

(In the last equality we use the assumption that the 1st type error does not occur, so $|\text{Int}_j \setminus I_j''| \leq (1 - \alpha + \delta)k$.) We set the length of the hash function hash_l^B to

$$L_B = \log S_B + r = (1 - \lambda)k + h(\alpha)\lambda k + \kappa_1 \delta \cdot k + \kappa_2 \log k + r$$

(here we choose suitable values of parameters κ_1 and κ_2). Hence, from Proposition 4 it follows that the probability of the 3rd type error is at most $1/2^r$.

We say that a block Y_j is *reconstructible*, if the errors of type 1, 2, and 3 do not occur for this j . For each block Y_j'' , probability to be non-reconstructible is at most $p_1^B + p_2^B + p_3^B$. This sum can be bounded by some threshold $\mu_B = \mu_B(n)$, where $\mu_B(n) \rightarrow 0$. For the chosen parameters $\delta(n)$ and $r(n)$ we have $\mu_B(n) = 1/(\log n)^c$ for some $c > 0$.

Since for each $j = 1, \dots, m$ the probability that Y_j'' is non-reconstructible is less than μ , we conclude that the expected number of non-reconstructible blocks is less than μm . This is already good news, but we need a stronger statement — we want to conclude that with high probability the number of non-reconstructible blocks is not far above the expected value.

Since random permutations in the construction are $(m^c \cdot k)$ -wise almost independent and the indices of hash functions are m^c -independent, we can apply Proposition 5 and Lemma 4(b). We obtain

$$\text{prob}[\text{the fraction of non-reconstructible blocks is greater than } 3\mu_B] = O(2^{-m^c})$$

for some $c > 0$.

We conclude that on stage (2_C) with probability $1 - O(2^{-m^c})$ Charlie decodes all blocks of Y_j'' except for at most $3\mu_B(n) \cdot m$ of them.

(3_C) Here Charlie reconstructs the string Y'' , if the number of non-reconstructible blocks Y_j'' (at the previous step) is less than $3\mu_B(n) \cdot m$. Indeed, $3\mu_B(n) \cdot m$ is just the number of errors that can be corrected by the Reed-Solomon checksums. Hence, the probability of failure at this step is less than $O(2^{-m^c})$. Here we choose the value of σ : we let $\sigma = 3\mu$.

Steps (4_C) and (5_C) : No errors.

Step (6_C) is similar to step (2_C) . We need to estimate the probabilities of errors in the reconstruction procedures for each block X_j' .

1st type error: the number of Alice's bits $x_{i_1}, \dots, x_{i_{\lambda n}}$ is less than $(\lambda - \delta)k$. (We cannot guarantee a correct reconstruction of a block X_j' if there are too few bits from Alice in this slot). We denote the probability of this event by p_1^A . From Lemma 4(a) it follows that $p_1^A \rightarrow 0$ since $\delta = 1/k^{0.49} \gg 1/\sqrt{k}$.

2nd type error: 1st type error does not occur but

$$\text{dist}(X'_{\text{Int}_j \setminus I_j'}, Y'_{\text{Int}_j \setminus I_j'}) > (\alpha + \delta)|\text{Int}_j \setminus I_j'|.$$

Denote the probability of this event by p_2^A . Again, from Lemma 4(a) it follows that $p_2^A \rightarrow 0$ since $\delta \gg 1/\sqrt{k}$.

3rd type error: 1st and 2nd type errors do not occur but there exist at least two different strings W satisfying (Cond_3) and (Cond_4) . All the positions Int_j are split into two classes: the set I_j' and its complement $\text{Int}_j \setminus I_j'$. For each position in I_j' Charlie knows the corresponding bit from X' sent by Alice. For the other bits Charlie already knows the bits of Y_j' , but not the bits of X_j' . To obtain Z , we should invert at most $(\alpha + \delta) \cdot |\text{Int}_j \setminus I_j'|$ bits of $Y'_{\text{Int}_j \setminus I_j'}$. The number of such candidates is equal to

$$S_A := \sum_{s=0}^{(\alpha+\delta)|\text{Int}_j \setminus I_j'|} \binom{|\text{Int}_j \setminus I_j'|}{s} \cdot 2^{|\text{Int}_j \setminus I_j'|} = 2^{h(\alpha)(1-\lambda)h(\alpha)k + O(\delta)k + O(\log k)}.$$

We set the length of the hash function hash_l^A to

$$L_A = \log S_A + r = (1 - \lambda)h(\alpha)\lambda k + \kappa_1 \delta \cdot k + \kappa_2 \log k + r$$

From Proposition 4 it follows that the probability of the 2nd type error $p_3^A \leq 1/2^r$.

We say that block X_j' is *reconstructible*, if the errors of type 1, 2, and 3 do not happen. For each block X_j' , probability to be non-reconstructible is at most $p_1^A(j) + p_2^A(j) + p_3^A(j)$. This sum is less than some threshold $\mu_A = \mu_A(n)$, where $\mu_A(n) \rightarrow 0$. For the chosen values of parameters we have $\mu_A(n) = 1/(\log n)^c$ for some $c > 0$.

Since the random permutations in the construction are $(m^c \cdot k)$ -wise almost independent and the indices of hash functions are m^c -independent, we get from Proposition 5 and Lemma 4

$$\text{prob}[\text{the fraction of non-reconstructible blocks is greater than } 3\mu_A] = O(2^{-m^c}).$$

Thus, with probability $1 - O(2^{-m^c})$ Charlie decodes on this stage all blocks of X_j' except for at most $3\mu_A \cdot m$ of them.

Step (7_C) is similar to step (3_C). At this step Charlie can reconstructs X' if the number of non-reconstructible blocks X_j' (at the previous step) is less than $3\mu_A \cdot m$ (this is the number of errors that can be corrected by the Reed-Solomon checksums). Hence, the probability of failure at this step is less than $O(2^{-m^c})$.

Step (8_C): No errors at this step.

Thus, with probability $1 - O(2^{-m^d})$ (for some $d < c$) Charlie successfully reconstructs strings X and Y .

V. CONCLUSION

Practical implementation. The coding and decoding procedures in our protocol run in polynomial time. However, the protocol does not seem very practical (mostly due to the use of the KNR generator from Proposition 2, which requires quite sophisticated computations). A simpler and more practical protocol can be implemented if we substitute t -wise almost independent permutations (KNR generator) by 2-independent permutation (e.g., a random affine mapping). The price that we pay for this simplification is a weaker bound for the probability of error, since with 2-independent permutations we have to employ only Chebyshev's inequality instead of stronger versions of the law of large numbers (applicable to n^c -wise almost independent series of random variables). (A similar technique was used in [8] to simplify the protocol from [6].) In this "simplified" version of the protocol we can conclude that the probability of error $\varepsilon(n)$ tends to 0, but the convergence is rather slow.

Another shortcoming of our protocol is very slow convergence to the asymptotically optimal communication complexity (the $o(\cdot)$ -terms in Theorem 4 are not so small). This is a general disadvantage of the concatenated codes and allied techniques, and there is probably no simple way to improve our construction.

Open problem: Characterize the set of all achievable pairs of rates for deterministic communication protocols.

REFERENCES

- [1] D. Slepian and J.K. Wolf. *Noiseless Coding of Correlated Information Sources*. IEEE Transactions on Information Theory, 19, 471–480 (1973).
- [2] A.D. Wyner, *Recent results in the Shannon theory*, IEEE Trans. Inf. Theory, vol. IT-20, no. 1, pp. 2-10, Jan. 1974.
- [3] Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. In proc. TCC 2004, pp. 446–472, 2004.
- [4] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. SIAM J. Discrete Math., 8(2):223–250, 1995.
- [5] E. Kaplan, M. Naor, and O. Reingold. *Derandomized construction of k -wise (almost) independent permutation*. Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques. 354–365 (2005).
- [6] A. Smith, *Scrambling Adversarial Errors Using Few Random Bits*, Optimal information reconciliation, and better private codes. In Proc. 18th ACM-SIAM Symposium on Discrete Algorithms (SODA), 395–404 (2007).
- [7] V. Guruswami and A. Smith, Codes for Computationally Simple Channels: Explicit Constructions with Optimal Rate, In Proc. 51st IEEE Symposium on Foundations of Computer Science (FOCS), 723–732 (2010).
- [8] A. Chuklin, *Effective protocols for low-distance file synchronization*, arXiv:1102.4712 (2011).
- [9] D. Chumbalov, *Combinatorial Version of the Slepian-Wolf Coding Theorem for Binary Strings*, Siberian Electronic Mathematical Reports, 10, 656–665 (2013).
- [10] F.J. MacWilliams, N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, 1977.
- [11] L. Carter, M. Wegman, *Universal Hash Functions*, Journal of Computer and System Science, 18,143–154, 1979.
- [12] A. y, K. Viswanathan, *One-way communication and error-correcting codes*. IEEE Transactions on Information Theory, 49(7), 1781–1788 (2003).
- [13] A. N. Kolmogorov. Three approaches to the quantitative definition of information. Problems of information transmission, 1(1), 1–7 (1965).
- [14] R. V. L. Hartley. Transmission of information. Bell System technical journal, 7(3), 535–563 (1928).
- [15] D. Hammer, A. Romashchenko, A. Shen, and N. Vereshchagin, Inequalities for Shannon entropy and Kolmogorov complexity. Journal of Computer and System Sciences, 60(2), 442–464 (2000).
- [16] A. Romashchenko, A. Shen, and N. Vereshchagin. Combinatorial interpretation of Kolmogorov complexity. Theoretical Computer Science, 271(1-2), 111–123 (2002).
- [17] H.-L. Chan. A combinatorial approach to information inequalities, Communications in Information and Systems, 1(3), 1–14 (2001).
- [18] An. Muchnik, Conditional complexity and codes, Theoretical Computer Science, 271(1-2), 97–109 (2002).
- [19] I. Csiszar and J. Körner. Information theory: coding theorems for discrete memoryless systems. 2nd ed. Cambridge University Press (2011).
- [20] V. Guruswami. List decoding of error-correcting codes. Springer Science & Business Media, 2004.
- [21] L.A. Bassalygo. New upper bounds for error-correcting codes. Problems of Information Transmission, 1 (1), 32–35 (1965).

APPENDIX

For the convenience of the readers and for the sake of self-containment, we proof in Appendix two results that appeared implicitly in preceding papers, see the discussion in Section II.

Proof of Proposition 1: To prove the proposition, it is enough to show that for every $\lambda \in (0, 1)$ there exists a linear encoding scheme with messages of length

$$\begin{aligned} m_A &= (\lambda + h(2\alpha))n + o(n), \\ m_B &= (1 - \lambda + h(2\alpha))n + o(n). \end{aligned}$$

We use the idea of syndrome encoding that goes back to [2]. First of all, we fix linear code with codewords of length n that achieves the Gilbert–Varshamov bound. Denote $H = (h_{ij})$ ($i = 1 \dots n$, $j = 1 \dots k$ for $k \leq h(2\alpha)n + o(n)$) the parity-check matrix of this code. So, the set of codewords $Z = z_1 \dots z_n$ of this code consists of all solutions of the system of uniform equations

$$\begin{cases} h_{11}z_1 + h_{12}z_2 + \dots + h_{1n}z_n = 0, \\ h_{21}z_1 + h_{22}z_2 + \dots + h_{2n}z_n = 0, \\ \dots \\ h_{k1}z_1 + h_{k2}z_2 + \dots + h_{kn}z_n = 0 \end{cases}$$

(a linear system over the field of 2 elements). W.l.o.g. we assume that the rank of this system is equal to k , and the last k columns of H make up a minor of maximal rank (if this is not the case, we can eliminate the redundant rows and re-numerate the columns of the matrix).

Remark: The assumption above guarantees that for every sequence of binary values h_1^0, \dots, h_k^0 and for any z_1, \dots, z_{n-k} we can uniquely determine z_{n-k+1}, \dots, z_n satisfying the linear system

$$\begin{cases} h_{11}z_1 + h_{12}z_2 + \dots + h_{1n}z_n = h_1^0, \\ h_{21}z_1 + h_{22}z_2 + \dots + h_{2n}z_n = h_2^0, \\ \dots \\ h_{k1}z_1 + h_{k2}z_2 + \dots + h_{kn}z_n = h_k^0 \end{cases}$$

(in other words, z_1, \dots, z_{n-k} are the free variables of this linear system, and z_{n-k+1}, \dots, z_n are the dependent variables).

Now we are ready to describe the protocol. Denote $s = \lfloor \lambda \cdot (n - k) \rfloor$.

Alice's message: given $X = x_1 \dots x_n$, send to Charlie the bits x_1, x_2, \dots, x_s and the syndrome of X , i.e., the product $H \cdot X^\perp$.

Bob's message: given $Y = y_1 \dots y_n$, send to Charlie the bits $y_{s+1}, y_{s+2}, \dots, y_{n-k}$ and the syndrome of Y , i.e., the product $H \cdot Y^\perp$.

Let us show that Charlie can reconstruct X and Y given these two messages. First of all, given the syndromes $H \cdot X^\perp$ and $H \cdot Y^\perp$, Charlie obtains $H \cdot (X + Y)^\perp$, which is the syndrome of the bitwise sum of X and Y . Since the distance between X and Y is not greater than αn , the bitwise sum $X + Y$ contains at most αn ones. The code defined by matrix H corrects αn errors, so Charlie can reconstruct all bits of $X + Y$ given the syndrome $H \cdot (X + Y)^\perp$.

Now Charlie knows the positions i where $x_i \neq y_i$, and the bits $x_1 x_2 \dots x_s$, and $y_{s+1} y_{s+2} \dots y_{n-k}$. This information is enough to reconstruct the first $n - k$ bits in both strings X and Y . Further, given the first $(n - k)$ bits of X and Y and the syndromes of these strings, Charlie reconstructs the remaining bits of X and Y (see the remark above).

The following proposition was implicitly proven in [12].

Proposition 8: For every small enough $\alpha > 0$ there exists a $\delta > 0$ such that for all large enough n , for the deterministic combinatorial Slepian–Wolf schemes with parameters (α, n) there is no achievable pairs of rates (m_A, m_B) in the (δn) -neighborhoods of the points $(n, h(\alpha)n)$ and $(h(\alpha)n, n)$ (points P_A and P_B in Fig. 4).

Proof: At first we remind the argument from [12, theorem 2]. It concerns an asymmetric version of the Slepian–Wolf scheme and it proves a lower bound for the length of $Code_A(X)$ assuming that $Code_B(Y) = Y$. Here is the idea of the proof: for each value c_A , the set of pre-images $Code_A^{-1}(c_A)$ is a set of strings with pairwise distances greater than $2\alpha n$, i.e., these pre-images make up an error correcting code that corrects αn errors. So we can borrow from the coding theory a suitable bound for the binary codes and use it to bound the number of pre-images of c_A . Then, we obtain a lower bound for the number of values of $Code_A(X)$ and, accordingly, for the length of $Code_A(X)$. Technically, if we know from the coding theory that for a binary code that corrects αn errors the number of codewords cannot be greater than $(1 - F(\alpha))n$ for some specific function $F(\alpha)$, then this argument implies that the length of $Code_A(X)$ cannot be less than $F(\alpha)n$. For example, in the well known Elias–Bassalygo bound $F(\alpha) = h(\frac{1}{2} - \frac{1}{2}\sqrt{1 - 2\alpha})$, which is stronger than the trivial volume bound $(1 - h(\alpha))n$, [21]. Alternatively, we can take the McEliece–Rodemich–Rumsey–Welch bound.

Though this argument deals with only very special type of schemes where $Code_B(Y) = Y$, it also implies some bound for the general Slepian–Wolf problem. Indeed, assume there exists a deterministic Slepian–Wolf scheme for string X and Y of

length n with $\text{dist}(X, Y) \leq T$ for some threshold $T = \alpha n$. Denote the lengths of Alice's and Bob's messages by

$$\begin{aligned} m_A &= (h(\alpha) + \delta_2)n, \\ m_B &= (1 - \delta_1)n \end{aligned}$$

respectively. We will prove that the pair of parameters (δ_1, δ_2) cannot be too close to zero. Notice that strings X and Y of any length $n' < n$ can be padded (by a prefix of zeros) to the length n . Hence, the given communication scheme (originally used for pairs of strings of length n , with the Hamming distance T) can be used also for the Slepian–Wolf problem with shorter strings of length $n' = (1 - \delta_1)n$ and the same distance between words T (which can be represented as $T = \alpha' n'$ for $\alpha' = \frac{\alpha}{(1 - \delta_1)}$). Thus, for the Slepian–Wolf problem with parameters (n', α') we have a communication scheme with messages of the same lengths m_A and m_B , which can be represented now as

$$\begin{aligned} m_A &= \frac{h(\alpha) + \delta_2}{1 - \delta_1} n', \\ m_B &= n'. \end{aligned}$$

We apply the explained above Orlitsky–Viswanathan bound to this scheme and obtain

$$\frac{h(\alpha) + \delta_2}{1 - \delta_1} \geq F\left(\frac{\alpha}{(1 - \delta_1)}\right)$$

(for any suitable bound $F(x)$ from the coding theory). It follows that

$$\delta_2 \geq (1 - \delta_1)F\left(\frac{\alpha}{(1 - \delta_1)}\right) - h(\alpha). \quad (5)$$

The functions $F(x)$ from the Elias–Bassalygo bound and from the McEliece–Rodemich–Rumsey–Welch bound are a continuous functions, and for small positive x they are bigger than $h(x)$. Hence, (5) implies that for every fixed α the values of δ_1 and δ_2 cannot be very small simultaneously.

We do not discuss here the exact shape of this forbidden zone for values of (δ_1, δ_2) ; we only conclude that small neighborhoods around the point $(h(\alpha)n, n)$ and (from a symmetric argument) $(n, h(\alpha)n)$ cannot be achieved, which concludes the proof of the proposition.